VIRTUAL FILE SYSTEM

FIELD OF THE INVENTION

5   The present invention relates to a virtual file system and in
    particular to a virtual file system for mobile devices for
    use with a mobile communications network.

BACKGROUND OF THE INVENTION

10

    One of the growth areas for mobile network operators and
    content providers is the provision of ringtones, wallpapers
    and other multimedia content for mobile telephones and
    devices.   There is a tension between the needs of mobile
15 network operators and device manufacturers to retain control
    over some aspects of the device user interfaces for branding
    purposes and the needs of users to customise and modify the
    appearance of their devices to suit their own needs.   The
    sophisticated software required to provide the desired
20 flexibility and customisation is also in tension with the
    limited processing power and data storage capacity of typical
    mobile devices.   The present invention seeks to mitigate
    these problems.

25 SUMMARY OF THE INVENTION

    According to a first aspect of the present invention there is
    provided a device comprising a storage means for storing a
    plurality data resources, a file system for organising the
30 plurality data resources stored in the storage means and a
    user interface for providing user access to the plurality
    data resources, wherein the file system comprises one or more

locations comprising one or more locations comprising
directly addressable data resources and one or more locations
comprising indirectly addressable data resources, the
indirectly addressable data resources being accessible
through a data provider and being configured, in use, to
provide a single interface from the user interface to both
directly addressable data resources and indirectly
addressable data resources.

The directly addressable data resources may comprise data
content files which, in use, are displayed within the user
interface. The indirectly addressable data resources may
comprise a database and, in use, the result of one or more
queries is displayed within the user interface. Furthermore,
the indirectly addressable data resources comprise a mark-up
language element and, in use, the mark-up language element is
rendered and the associated result is displayed within the
user interface.

According to a second aspect of the present invention there
is provided a method of for storing a plurality of data
resources within a file system of a device, the method
comprising the steps of: defining one or more locations
comprising one directly addressable data resources; defining
one or more locations comprising indirectly addressable data
resources, the indirectly addressable data resources being
accessible through a data provider; wherein file system
provides a single interface from the user interface to access
both the directly addressable data resources and indirectly
addressable data resources access.

According to a third aspect of the present invention there is

provided a data carrier comprising computer executable code
for performing the above-described method(s).

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a schematic depiction of a system
incorporating the present invention;

Figure 2 depicts in greater detail the structure and
operation of server 100;

Figure 3 shows a schematic depiction of the software 400
for the mobile devices 300;

Figure 4 shows a schematic depiction of the content
toolset 200;

Figure 5 shows a schematic depiction of four hierarchical
planes; and

Figure 6 shows a schematic depiction of a device 800 that
comprises a user interface according to an embodiment of
the present invention

DETAILED DESCRIPTION OF THE INVENTION

The invention will now be described by way of illustration
only and with respect to the accompanying drawings, in which
Figure 1 shows a schematic depiction of a system
incorporating the present invention. The system comprises
server 100, content toolset 200, mobile devices 300,
operational support systems (OSSs) 700, content feeds 500 and
user interface (UI) sources 600. In use, the server 100
communicates content data and UI data to the mobile devices
300, 301, …, each of which comprise software package 400.
The server 100 interfaces with OSSs 700, with the OSSs being
those conventionally used to operate mobile networks, for

example billing, account management, etc.   The server 100
further interfaces with the content toolset 200: the content
toolset receives data from UI sources 600, 601, …, and
packages the UI data such that the server can transmit the
5   packaged UI data to the software packages 400 comprised
within the mobile devices 300. The server receives data from
a plurality of content feeds, and this data is processed and
packaged such that it can be sent to the software packages
400 or so that the mobile devices 300 can access the data
10 using the software package 400.

The system can be envisaged as being divided into three
separate domains: operator domain 50 comprises the systems
and equipment operated by the mobile network operator (MNO);
15 user domain 60 comprises a plurality of mobile devices and
third-party domain 70 comprises the content feeds and UI
feeds that may be controlled or operated by a number of
different entities.

20 Figure 2 depicts in greater detail the structure and
operation of server 100.   Server 100 comprises publishing
component 110 and content server component 150. Publishing
component comprises database 111, import queue 112, content
toolset interface 113, user interface 114 & catalogue 115.
5   In operation, the publishing component receives content from
the content toolset at the content toolset interface.   The
content is presented in the form of a parcel 210a, 210b, …,
(see below) comprising one or more Trigs and one or more
Triglets.   A trig is a user interface for a mobile device,
]   such as a mobile telephone and a triglet is a data file that
can be used to extend or alter a trig. If a parcel comprises
more than one trig then one of the Trigs may be a master trig

from which the other Trigs are derived.

The publishing component user interface 114 can be used to import a parcel into the database 111, and this process
5   causes references to each trig and triglet to be loaded into the import queue 114, which may comprise references to a plurality of parcels 210a, 210b,… .   The contents of the parcel may be examined using the user interface and the contents of the parcel can be passed to the catalogue.
10

Figure 3 shows a schematic depiction of the software 400 for the mobile devices 300, which comprises a mark-up language renderer 410, update manager 420, network communication agent 425, resource manager 430, virtual file system 435, actor
15 manager 440, a plurality of actors 445a, 445, …, native UI renderer 450, support manager 460, trig manager 465 and mark-up language parser 470.

It is preferred that the software operates using TrigML,
20 which is an XML application and that mark-up language renderer 410 renders the TrigXML code for display on the mobile device 300. The mark-up language renderer also uses the TrigML Parser to parse TrigML resources, display content on the device screen and controlling the replacement and
25 viewing of content on the handset. The native UI renderer is used to display UI components that can be displayed without the use of TrigML, and for displaying error messages.

The software 400 is provisioned and installed in a device
30 specific manner. For example for a Nokia Series 60 device the software is installed using a SIS file, whereas for a MS Smartphone device the software is installed using a CAB file.

Similarly, software upgrades are handled in a device specific
manner. The software may be provisioned in a more limited
format, as a self-contained application that renders its
built in content only: i.e. the software is provisioned with
5 a built-in trig but additional trigs cannot be added later.
The supplied trig may be upgraded over the air.

The trig manager 465 presents an interface to the resource
manager 430 and the mark-up language renderer. It is
10 responsible for trig management in general. This includes:
persisting knowledge of the trig in use, changing the current
trig, selection of a trig on start-up, selection of a further
trig as a fall back for a corrupt trig, maintaining the set
of installed trigs, identifying where a particular trig is
15 installed to the resource manager and reading the update
channel definitions of a trig and configuring the update
manager appropriately.

The resource manager provides an abstraction of the
20 persistent store on device, i.e. storing the files as real
files, or as records in a database. The resource manager
presents a file system interface to the mark-up language
renderer and the update manager. It is responsible for
handling file path logic, distinguishing between real
25 resource files and actor attributes, mapping trig-relative
paths onto absolute paths, interfacing with the trig manager
and providing a modification interface to the update manager.

The Resource Manager is also responsible for ensuring the
30 integrity of the resources stored in the persistent store,
especially in the face of unpredictable interruptions such as
loss of device power. The Resource Manager has no knowledge

of the trig currently used. Its interface is thread safe (as
it may be used by both the Update Manager and the Renderer
from different threads.

5   The Update Manager handles the reception and application of
     Trigs and Triglets. The Update Manager presents an interface
     to the Renderer and the trig Manager and is responsible for:
     the initiation of manual updates when instructed to by the
     Renderer; controlling and implementing the automatic update
10   channel when so configured by the trig manager; indicating
     the progress of a manual update and recovering an Update
     following unexpected loss of network connection and/or device
     power. The update packet format may be defined as a binary
     serialisation of an XML schema.

15

     The Support Manager provides an interface for other
     components to report the occurrence of an event or error.
     Depending on the severity of the error, the Support Manager
     will log the event and/or put up an error message popup

20

     XML is a convenient data formatting language that is used to
     define the update packet format as well as TrigML content.
     For bandwidth and storage efficiency reasons, text XML is
     serialised into a binary representation. Both update packets
25   and TrigML fragments are parsed by the same component, the
     mark-up language parser. Any further use of XML in the
     software must use the binary XML encoding and therefore re-
     use the parser.

30   The Actor Manager 440 looks after the set of actors 445
     present in the software. It is used by: the renderer when
     content is sending events to an actor; actors that want to

notify that an attribute value has changed and actors that
want to emit an event (see below).

The software may comprise a multi-threaded application
running a minimum of two threads, with more possible
depending on how many and what sort of actors are included.
The software runs mostly in one thread, referred to as the
main thread. The main thread is used to run the renderer
which communicates synchronously with other components.
Actors always have a synchronous interface to the Renderer.
If an actor requires additional threads for its
functionality, then it is the responsibility of the Actor to
manage the inter-thread communication. It is preferred that
a light messaging framework is used to avoid unnecessary code
duplication where many actors require inter-thread
communication.

In addition to the main thread, the update manager runs a
network thread. The network thread is used to download
update packets and is separate from the main thread to allow
the renderer to continue unaffected until the packet has
arrived. The Update Manager is responsible for handling
inter-thread messaging such that the Update Manager
communicates synchronously with the Renderer and Resource
Manager when applying the changes defined in an Update
Packet.

The memory allocation strategy of the software is platform
specific. On MIDP platforms, the software simply uses the
system heap and garbage collector for all its memory
requirements. Garbage collection is forced whenever a content
replacement event occurs in an attempt to keep the garbage

collection predictable and not suffer unexpected pauses in operation. It is assumed that any memory allocation might fail, in which case the software will delete all its references to objects, garbage collect, and restart – assuming that the software has already successfully started up and rendered the first page.

On C++-based platforms, a mixture of pre-allocation and on-demand allocation will be made from the system heap. All memory required for start-up is allocated on-demand during start-up, with any failures here causing the exit (with message if possible) of the software. Following successful start-up, memory needed for rendering the content document model is pre-allocated. Provided content is authored to use less than a defined limit, it is guaranteed to render. Additional use is made of RAM for various caches needed for fast operation of the software. Where memory conditions are low, these caches will be released resulting in slow rendering performance from the software.

Errors that are severe enough to disrupt the normal operation of the software must result in a pop-up dialog box. The dialog box contains one of a small number of internationalised error messages (internationalised versions of these strings may be compiled into the software at build-time with the version of an error string to display being determined by the relevant language setting on the device). To keep the number of messages to a minimum, only a few generic problems are covered.

To allow for support situations, error dialogs also display an error code as a 4-digit (16-bit) hex string. Each error

code is associated with a description text that can be used
by support staff to determine the nature of a problem with
the software. Errors that occur in the software and that are
not severe enough to halt its operation may be logged by the
5   Support Manager component. The Support Manager can be queried
by the user typing special key sequences. The Support Manager
can also supply its error log to a server via an HTTP GET or
POST method.

10 The Renderer receives information regarding the key press. If
there is no behaviour configured at build time for a key, it
is sent as a TrigML content event to the current focus
element. The content event is then handled as defined by
TrigML's normal event processing logic.

15

For example, if a key is pressed down, a 'keypress' event is
delivered to the Renderer with a parameter set to they
relevant key. When the key is released, a '!keypress' event
is delivered to the Renderer. If a key is held down for a
20 extended period of time, a 'longkeypress' event is delivered
to the renderer. On release, both a '!longkeypress' and a
'!keypress' event are delivered to the Renderer.

Whenever the software is started, it executes the following
25 actions:

    • Check for, and continue with, interrupted Update
      processing;

    • Check for, and process, Updates residing in the file
      system (either pre-provisioned, or installed to the file
30   system by some other means);

    • If known, start the current trig (which may be the last
      run trig);

- If a current trig is not set, a trig that has been flagged as a 'default' trig can be started.
- Failing the presence of a default trig, the first valid trig by alphabetical order of name will be selected.

A trig is started by loading the defined resource name, start-up/default. The TrigML defined in start-up/default is parsed as the new contents for the content root node.

The first time a trig is run by the software following its installation, the trig is started by loading the resource name startup/firsttime. The software may record whether a trig has been run or not in a file located in the top level folder for that trig. Dependent on the platform used by the mobile device, the automatic start-up of the software may be set as a build-time configuration option. Furthermore, placing the software in the background following an auto-start may also be a build-time configuration option.

A launcher may appear to the user as an application icon and selecting it starts the software with a trig specified by that launcher (this trig may be indicated by a launcher icon and/or name). When using a launcher to start a trig, it is possible to specify an 'entry point' parameter. The parameter is a resource name of a file found in the 'start-up' folder. This file is not used if the trig has never been run before, in which case the file called 'firsttime' is used instead.

The software uses content resource files stored in a virtual file system on the device. The file system is described as virtual as it may not be implemented as a classical file-system, however, all references to resources are file paths

as if stored in a hierarchical system of folders and files.

Details regarding the arrangement of the file-system are given below in Appendix A. Furthermore, the software stores
5   some or all of the following information: usage statistics; active user counts; TrigManager state; TrigML fragments & update channel definition (serialised as binary XML); PNG images; plain text, encoded as UTF-8 OTA and then stored in a platform specific encoding; other platform specific
10 resources, e.g. ring tone files, background images, etc.

Files in the file system can be changed, either when an actor attribute value changes, or when a file is replaced by a triglet. When files in the /attrs directory change, the
15 Renderer is immediately notified and the relevant branches of the content tree are updated and refreshed. When images and text resources are changed, the Renderer behaves as if the affected resources are immediately reloaded (either the whole content tree or just the affected branches may be refreshed).
20 When TrigML fragments are changed, the Renderer behaves as if it is not notified and continues to display its current, possibly out of date, content. This is to avoid the software needing to persist <include> elements and the <load> history of the current content.
25
The software 400 is provisioned to mobile devices in a device specific method. One or more Trigs can be provisioned as part of the installation, for example, stored as an uncompressed update packet. On start-up, the packet can be
30 expanded and installed to the file-system.

The Actors 445 are components that publish attribute values

and handle and emit events. Actors communicate with the
Renderer synchronously. If an actor needs asynchronous
behaviour, then it is the responsibility of the actor to
manage and communicate with a thread external to the main
5   thread of the Renderer.

Actor attributes may be read as file references. Attributes
are one of four types: a single simple value; a vector of
simple values; a single structure of fields, each field
10 having a simple value; or a vector of structures. Attributes
may be referenced by an expression using an object.member
notation similar to many object-orientated programming
languages:

15   <image res="signallevels/{protocol.signalstrength}"/>

When needed as a file, an attribute is accessed via the
/attrs folder.

20   <text res="/attr/network/name">

An Actor can be messaged by sending it an event with the
<throw> element. Events emitted by actors can be delivered
to the content tree as content events: these can be targeted
25 at an element Id or 'top'. The interface to an actor is
defined by an Actor Interface Definition file. This is an XML
document that defines the attributes, types, fieldnames,
events-in and parameters, and events out. The set of actors
is configurable at build-time for the software. Appendix B
30 gives an exemplary listing of some actors that may be used,
along with the associated functions or variables.

One of the limitations that is common in most mobile devices
is that the display screen is quite small and when a menu is
displayed it is not always possible to display all of the
menu items on the screen at one time. Conventional
5   approaches tend to load all of the menu items into memory,
along with associated icons or graphics, and then display
them appropriately as the user scrolls up or down the menu.

There is provided a technique that limits the number of menu
10 items to be loaded into memory to the number of items that
can be displayed on the screen at a time. When the user
scrolls along the menu, the item(s) no longer on display are
discarded and the item(s) now on display are loaded into
memory.

15

Preferably, this can be implemented by using a <griddata>
element in TrigML to define a list view of some data, where
the data is stored in a folder in the file system, and the
list appearance has the same structure for each item. The
20 <griddata> element comprises a 'repeat-over' attribute that
specifies the folder in which the data can be located. The
single child element of <griddata> is a template for the
appearance of each item in the list.

25 The template uses a special symbol, e.g. '$$' to refer to the
iterator. This is the template variable that changes each
time the template is instantiated: for example

```
    <griddata repeatover="news/headlines">
0        <text res="news/headlines/$$/title.txt"/>
    </griddata>
```

where the folder news/headlines/ contains:

```
0/title.txt
1/title.txt
2/title.txt
3/title.txt
```

This would display a list of 4 items, each described by a simple <text> element pointing the 'title.txt' resource in the 'news/headlines/$$' folder. Where the source data has more items in it than the <griddata> element has room for on the display, the <griddata> element only displays those items that can be displayed. When the user scrolls through the list, the <griddata> element shifts the 'data-window' accordingly. The advantage of this technique is that only the resources required by the current display are actually loaded in memory, which reduces the memory utilisation and reduces the amount of time taken to render the list of items.

A similar scheme can be used to define the order that a list is displayed in. If the target of the 'repeat-over' attribute is a file instead of a folder, then the file can be assumed to contain a list of resource names to use in the iteration. For example,

```
<griddata repeatover="football/league">
    <text res="football/teams/$$/name.txt"/>
</griddata>
```

where the file football/league contains:

```
Manchester
Arsenal
Chelsea
```

the folder football/teams/ contains:

        Manchester/name.txt
        Arsenal/name.txt
        Chelsean/name.txt

5

and each name.txt is a text file holding the team name. The
result of this is that the test files associated with the
teams would be displayed in the defined order and within the
defined area of the device display.

10

The software utilise a virtual file system 435, which allows
a conventional file system to be incorporated with other
forms of data such that data stored in any part of the
virtual file system can be accessed in a common fashion.

15 Examples of non-conventional data storage that can be
integrated within such a virtual file system include data
stored within a database or data that it generated on demand
by a further software component.

20 For example, in conventional mobile devices, information
regarding the battery strength, signal strength, new text
messages, etc. are shown to the use and this information is
typically obtained by the operating system sending a call to
the relevant hardware or software entity and the UI
25 interpreting the received answer and displaying it.

This information may be displayed in the UI using a TrigML
tag (see below), such as <phonestatus> (or <signalstrength>).
Rendering this tag causes a listening query to be opened to
30 the relevant hardware or software entity. If a change in
state occurs then the UI renderer is notified and the UI
renderer loads the relevant icon or graphic to communicate

the change in state to the user.   If the user changes the
view within the UI the tag may be withdrawn and the listening
query is terminated.   This approach is more resource
efficient as the listening query is only active when the tag
5   is in use.

The virtual file system comprises a single root which
contains both directly addressable data resources, such as
content files (for example, text, graphics, video, audio,
10   etc.), and indirectly addressable data resources, such as
TrigML tags or data stored within a database. This provides
a user with a unified view through the user interface that
enables the user to access the different types of data held
within the virtual file system.   Whilst the different data
15   types are presented to the user in this manner, the data can
still be stored in a method that enables efficient data
storage and retrieval.

For commercial reasons it is desirable for MNOs and/or
20   content providers to be able to have some control over the
user interface that will be displayed on the screen of a
mobile device.   It is also important that there is a degree
of flexibility that allows users to download triglets or new
trigs to modify the appearance of their device and also to
25   make further changes to the displayed image that is
determined by the trig or triglet in use.

This problem can be addressed by considering the UI to be
formed from a number of hierarchical planes, each of the
30   levels comprising one or more entities of the UI.   By
assigning a hierarchy to the MNO, device manufacturer, trig
provider and the device user it is possible to provide the

required levels of permission.

Figure 5 shows a schematic depiction of four hierarchical
planes 405a-d: plane 405a comprises UI elements defined buy
5   the MNO; plane 405b comprises UI elements defined buy the
device manufacturer; plane 405c comprises UI elements defined
by a trig; and plane 405d comprises UI elements defined buy
the user. Plane 405a has the highest position in the
hierarchy and plane 405d has the lowest position in the
10 hierarchy. For example, the mno_logo element in plane 405a
defines the graphic element used and its position on the
display screen of the device. As it is in the highest plane
of the hierarchy it will always appear and will take
preference over any other UI element in a lower hierarchy
15 element that attempts to use the pixels used by mno_logo.
Plane 405d comprises the backgroundcolour element, which is
not defined in any of the other planes and thus the colour
defined in backgroundcolour will be used in the UI.

20 Plane 405c comprises the windowtitle.txt element that defines
the attributes for the text used in the title of a window.
This may be overwritten by adding a windowtitle.txt element
to either plane 405a or 405b to define the text attributes,
or by adding a windowtitle.txt_deleted element to either
25 plane 405a or 405b to instruct the UI renderer to ignore any
subsequent windowtitle.txt element.

The user can set a preference within the software 300 to
control the content that is displayed within the UI. For
30 example, content relating to a number of football teams may
be stored on a server with a path having a form similar to

/demoUI/football/team_xxxx/team_menu.png

where the team_xxxx variable is selected by the user from a
list of teams (manu, chel, leed, manc, etc.) and inserted
into the path such that the UI displays the content related
to the selected team.   A change in the team_xxxx variable
will cause the content displayed to altered accordingly.   It
should be noted that the selection of a preference controls
the display of content that is selected from content stored
on a remote server, as opposed to selecting from content that
is stored on local storage.

This approach is preferable to sending a request of the form

http://t1.trigenix.com/triglets/football/triglet&pn="07766554
43322"

as in this case the server needs to perform a database query
in order to identify the content to be displayed and this
will significantly increase the resources required from the
server to provide the requested content.

Another known technique by which the same result can be
achieved is to send a request of the form:

http://t1.trigenix.com/triglets/football/triglet&fc='ManU'

but a disadvantage of this approach is that each time a new
team is added then the server logic must be updated to
include the new team.   In contrast, this method requires that
content is added to the server at a new location, which is a
simpler process and requires fewer resources to implement it.

Updates comprise a new trig (a new or replacement UI) or a
triglet (a modification to an existing trig) and may be
regarded as modifications to the software file-system. The
Update Manager to determine what needs changing in the file-
system by reading a packet. Update Packets may be downloaded
over the air by the software 400 using HTTP, or other
suitable transport mechanisms, wrapped in a device-specific
package format or pre-provisioned with the installation of
the software itself.

Updates may be triggered by a number of means, which include

   • the software checking for interrupted Update processing
     on start-up

   • the software checking for pre-installed Update Packets
     on start-up

   • automatically as configured by an Update Channel

   • user initiation

   • the device receiving a special SMS

The algorithm used to unpack and install an update is device
specific. However, it is important that the algorithm is safe
from unexpected interruption (e.g. power loss), such that no
corruption or unrecoverable state is reached in the file-
system. This may be achieved by using two threads (a network
thread and a renderer thread) with the goal of having as much
of the update processing as possible being performed by the
network thread so as to interrupt the renderer thread for the
shortest possible amount of time.

TrigML fragments are files containing text TrigML and
resource references inside these fragments are virtual file

paths. The mapping of these virtual file paths to real file
paths is defined by a TrigDefinition file. This file also
defines other properties of the trig. When used for compiling
a triglet, this file also defines how the input
5   TrigML/PNG/Text resources map onto modifications of the
virtual file-system of a trig. For PNG and Text Resources
the trig Definition file points at a list of real files on
the host file-system and the resources are copied to the
outputs.

10

TrigML can use constant variables instead of attribute
values. Constant variables are accessed with the same syntax
as <include> parameters, e.g. $background_colour. Constants
are treated as global variables in a trig and are defined in
15 the reserved folder, constants/. The variable definitions
contained in the files in the constants/ folder may be
resolved at compile time with direct substitution of their
values. Alternatively the variable definitions in constants/
are compiled as global variables and resolved at content
20 parse time by the software. This allows the trig to be
updated by a simple replacement of one or all of its
constants files.

A System String Dictionary defines the integer values to use
25 for all well known strings, i.e. reserved words. These have
several types, including: TrigML element and attribute names
('group', 'res', 'layer', 'image', 'x'), TrigML attribute
values (e.g.: 'left', 'activate', 'focus') and common
resource paths (e.g.: 'attr', 'start-up', 'default'). As an
30 input, the String Dictionary is optional. The first time a
trig is compiled it will not have a String Dictionary. This
first compilation creates the String Dictionary, which is

then used for all future compilations of that trig. Triglet
compilation must have a String Dictionary that defines all
the string mappings used by the trig it is modifying.

5   In order to successfully render the user interface of a
mobile device, the mark-up language must have the following
qualities: concise page definitions, consistent layout rules,
be implementable in a compact renderer, provide multiple
layering and arbitrary overlapping content, event model,
10 require the repaint of only the areas of the display that
have to change between pages of the UI, include hooks to the
platform for reading property values receiving events and
sending events, extensible, and be graphically flexible.
TrigML provides these features and our co-pending application
15 GB0403709.9, filed February 19th 2004, gives an overview of
the elements and attributes that provide the desired
functionality.

It is desirable that the cost of re-branding UIs and
20 producing a continual stream of updates is minimal. This is
enabled by providing an efficient flow of information from
the creative process through to the transmission of data to
users.

25 A container, referred to as a parcel, is used for UIs, UI
updates, and templates for 3rd party involvement. Parcels
contain all the information necessary for a 3rd party to
produce, test and deliver branded UIs and updates. Figure 4
shows a schematic depiction of the content toolset 200,
30 which comprises scripting environment 220, test and
simulation environment 230 and maintenance environment 240

The parcel process comprise five processing stages:

1) The scripting environment 220 provides the means to design the template for one or more UIs and the update strategy for UIs based on that template.

2) The maintenance environment 240 provides for rapid UI and update production in a well-controlled and guided environment that can be outsourced to content providers.

3) The maintenance environment 240 'pre-flight' functionality allows the deployment administrator to check and tune the UIs and updates that they receive from 3rd parties.

4) The publishing component 110 provides management of UIs and updates at the deployment point, including the staging of new releases.

5) The publishing component 110 enables the automatic generation of updates from live content feeds.

In a typical project, parcels are created within the scripting environment 220 for: a content provider to create re-branded UIs from a template, incorporating the same 'feel' but a different 'look'; a content provider to create updates from a template, that provide a periodic, or user selected variation to UI content ; or an ad agency to create updates from a template that promote new services on a periodic basis.

For all of these use cases, maintenance environment 240 is used to import the parcel, re-brand and reconfigure the content and create a new parcel for submission to the publishing component 110. In the design of the UI template, the following issues should be considered: which part of the UI can be re-banded; which features of a UI need to be reconfigured at re-branding or remotely; which part of the UI

content may be updated; and if the UI is re-branded then can
user select content feeds in use. The scripting environment
220 allows these strategies to be defined, and enables the
maintenance environment 240 as the implementer of each
5   instance of each strategy.

A parcel is generated by the scripting environment 220 which
comprises a template UI or update for editing. Once editing
is complete the parcel is saved in an 'outbox' ready for
10 despatch to the maintenance environment 240 for publishing to
the content server. The following 'parcel' functions are
provided.   The maintenance environment 240   can be used to
edit/replace resources held within the parcel.   Parcels can
be exported to the simulation environment to test the
15 performance of the UI or UI update on a mobile device.

Many different UIs can be derived from a common base.
Typically the common base would implement most of the
interface itself, and Trigs derived from it would implement
20 small variations on it, such as branding. A Triglet can be
derived from a Trig, and it can override any of the resources
from the parent Trig that it chooses to (optionally it may
introduce its own resources). Note that "resources" here also
refers to TrigML, so the behaviour and layout of a Trig can
25 be modified by a Triglet just as easily as it replacing a
single image or piece of text.

A Parcel may comprise one or more base Trigs (i.e. a Trig
that is not derived from any other trig), one or more
30 multiple Trigs derived from a base Trig, a plurality of
triglets derived from any of the trigs and a plurality of
triglets derived from other triglets.

The parcel format is an opaque binary format that stores all
this information as serialized objects.   The parcel may
comprise a number of resources , such as images, text, URLs,
5   update   channels,   ringtone   files,   wallpapers,   native
applications, etc.   Each resource contains permission
information as to how to view, edit, or delete the resource.
Each resource furthermore contains meta information such as
documentation and instructions that are relevant to that
10 resource. Each Parcel tool either assumes a relevant role, or
requires users to login as a particular role.

Figure 6 shows a schematic depiction of a device 800 that
comprises a user interface according to an embodiment of the
15 present invention.   The device comprises a display 810 that
displays the user interface 815 and user interface means 820,
that enable the user to interact with the user interface 815.
A processor 830 executes the software that is stored within
one or more storage means 840 and there may be provided one
20 or more wireless communication interfaces 850, to enable
communication with other devices and/or communication
networks. One or more batteries 860 may be received to power
the device, which may also comprise interfaces to receive
electrical power and/or communication cables.
25

The nature of these components and interfaces will depend
upon the nature of the device.   It will be understood that
such a user interface can be implemented within a mobile or
cellular telephone handset, but it is also applicable to
30 other portable devices such as digital cameras, personal
digital organisers, digital music players, GPS navigators,
portable gaming consoles, etc.   Furthermore, it is also

applicable to other devices that comprise a user interface,
such as laptop or desktop computers.

The user interface means may comprise a plurality of buttons,
such as a numerical or alpha-numerical keyboard, or a touch
screen or similar. One or more storage devices may comprise
a form of non-volatile memory, such as a memory card, so that
the stored data is not lost if power is lost. ROM storage
means may be provided to store data which does not need
updating or changing. Some RAM may be provided for temporary
storage as the faster response times support the caching of
frequently accessed data. The device may also accept user
removable memory cards and optionally hard disk drives may be
used as a storage means. The storage means used will be
determined by balancing the different requirements of device
size, power consumption, the volume of storage required, etc.

Such a device may be implemented in conjunction with
virtually any wireless communications network, for example
second generation digital mobile telephone networks (i.e.
GSM, D-AMPS), so-called 2.5G networks (i.e. GPRS, HSCSD,
EDGE), third generation WCDMA or CDMA-2000 networks and
improvements to and derivatives of these and similar
networks. Within buildings and campuses other technologies
such as Bluetooth, IrDa or wireless LANs (whether based on
radio or optical systems) may also be used.   USB and/or
FireWire connectivity may be supplied for data
synchronisation with other devices and/or for battery
charging.

Computer software for implementing the methods and/or for
configuring a device as described above may be provided on

data carriers such as floppy disks, CD-ROMS. DVDs, non-volatile memory cards, etc.

This application claims the benefit of UK Patent Application number 0403709.9, filed February 19th 2004, the contents of which are incorporated herein by reference.

APPENDIX A

For file paths beginning with a leading '/':

/attrs		Like the unix /proc directory, stores actor attribute values for reference by content when the attribute is needed as a file reference.
	<actor>	Each subdirectory of /attrs is the actor name.
	<attribute>	Each attribute is accessed as a node in the actor subdirectory.
	<field>	If the attribute is a structure, then the field name specifies which structure member to access.
	<index>	If the attribute is a vector attribute, then the index number specifies the index into the vector of the desired attribute.
	<field>	If the vector attribute is a collection of structures, then the field name again specifies the structure member.

File paths without a leading '/' are treated as relative to the current trig, i.e. every trig is stored in its own folder hierarchy rooted in a single folder.

config	Common folder in every trig to store trig meta data.
channels	Common folder to store the update channel definitions.
<channel defs>	Set of files defining the collection of update channels for the trig. Each file can define one or more update channels.
start-up	Common folder to store entry points for the trig.
default	Common TrigML file to store the default entry point for the trig.
firsttime	Common TrigML file to store the TrigML for use the first time this trig is run
<trigml files>	Other named TrigML files can be used as entry points if found in the start-up folder.
constants	This folder is not passed OTA and is instead fully resolved at content compile time.
<rest of content>	trig content is organised in trig-defined format under the Trigs folder.

APPENDIX B

Trigplayer	Attributes	UpdateState	
Actor	Messages	exit	
		predial_mode	on/off
	Events	idle	

Launch Actor	Attributes		
	Messages	browser	url
		SMS	Number
			message
		Camera	
		Inbox	
		Profiles	
		missed_calls	
		dialer	number
		...	
		native_app	app_id
			url
	Events		

5

Install Actor	Attributes		
	Messages	ringtone	resource_path
		wallpaper	resource_path
	Events		

Phone Actor	Attributes	Bluetooth	
		IrDA	
		Call	
		GPRS	
		UnreadSMS	
		UnreadVoiceMail	
		UnreadMsgs	
		BatteryLevel	
		SignalStrength	
	Messages		
	Events	missed_call	
		message_arrived	
		voice_mail_arrived	